autogluon_each_location

October 6, 2023

[66]:
```
import pandas as pd
from darts import TimeSeries
import numpy as np



import warnings
warnings.filterwarnings("ignore")

def fix_datetime(X, name):
    # Convert 'date_forecast' to datetime format and replace original column␣
 ↪with 'ds'
    X['ds'] = pd.to_datetime(X['date_forecast'])
    X.drop(columns=['date_forecast'], inplace=True, errors='ignore')
    X.sort_values(by='ds', inplace=True)
    X.set_index('ds', inplace=True)

    # Drop rows where the minute part of the time is not 0
    X = X[X.index.minute == 0]
    return X



def convert_to_datetime(X_train_observed, X_train_estimated, X_test, y_train):
    X_train_observed = fix_datetime(X_train_observed, "X_train_observed")
    X_train_estimated = fix_datetime(X_train_estimated, "X_train_estimated")
    X_test = fix_datetime(X_test, "X_test")


    X_train_observed["estimated_diff_hours"] = 0
    X_train_estimated["estimated_diff_hours"] = (X_train_estimated.index - pd.
 ↪to_datetime(X_train_estimated["date_calc"])).dt.total_seconds() / 3600
    X_test["estimated_diff_hours"] = (X_test.index - pd.
 ↪to_datetime(X_test["date_calc"])).dt.total_seconds() / 3600

    X_train_estimated.drop(columns=['date_calc'], inplace=True)
    X_test.drop(columns=['date_calc'], inplace=True)
```

```python
    y_train['ds'] = pd.to_datetime(y_train['time'])
    y_train.drop(columns=['time'], inplace=True)
    y_train.sort_values(by='ds', inplace=True)
    y_train.set_index('ds', inplace=True)

    return X_train_observed, X_train_estimated, X_test, y_train




def preprocess_data(X_train_observed, X_train_estimated, X_test, y_train,␣
 ↪location):
    # convert to datetime
    X_train_observed, X_train_estimated, X_test, y_train =␣
 ↪convert_to_datetime(X_train_observed, X_train_estimated, X_test, y_train)

    y_train["y"] = y_train["pv_measurement"].astype('float64')
    y_train.drop(columns=['pv_measurement'], inplace=True)
    X_train = pd.concat([X_train_observed, X_train_estimated])


    # clip all y values to 0 if negative
    y_train["y"] = y_train["y"].clip(lower=0)

    X_train = pd.merge(X_train, y_train, how="outer", left_index=True,␣
 ↪right_index=True)

    X_train["location"] = location
    X_test["location"] = location

    return X_train, X_test
# Define locations
locations = ['A', 'B', 'C']

X_trains = []
X_tests = []
# Loop through locations
for loc in locations:
    print(f"Processing location {loc}...")
    # Read target training data
    y_train = pd.read_parquet(f'{loc}/train_targets.parquet')

    # Read estimated training data and add location feature
    X_train_estimated = pd.read_parquet(f'{loc}/X_train_estimated.parquet')

    # Read observed training data and add location feature
```

```
    X_train_observed= pd.read_parquet(f'{loc}/X_train_observed.parquet')

    # Read estimated test data and add location feature
    X_test_estimated = pd.read_parquet(f'{loc}/X_test_estimated.parquet')

    # Preprocess data
    X_train, X_test = preprocess_data(X_train_observed, X_train_estimated,␣
 ↪X_test_estimated, y_train, loc)

    X_trains.append(X_train)
    X_tests.append(X_test)

# Concatenate all data and save to csv
X_train = pd.concat(X_trains)
X_test = pd.concat(X_tests)


# temporary
# X_train["hour"] = X_train.index.hour
# X_train["weekday"] = X_train.index.weekday
# X_train["month"] = X_train.index.month
# X_train["year"] = X_train.index.year

# X_test["hour"] = X_test.index.hour
# X_test["weekday"] = X_test.index.weekday
# X_test["month"] = X_test.index.month
# X_test["year"] = X_test.index.year

X_train.dropna(subset=['y'], inplace=True)
X_train.to_csv('X_train_raw.csv', index=True)
X_test.to_csv('X_test_raw.csv', index=True)
```

```
Processing location A…
Processing location B…
Processing location C…
```

[67]:
```
# auto.dataset_overview(train_data=X_train[X_train["location"]=="B"],␣
 ↪test_data=X_test[X_test["location"]=="B"], label="y", sample=None)
```

1 Starting

[68]:
```
import os


# Get the last submission number
last_submission_number = int(max([int(filename.split('_')[1].split('.')[0]) for␣
 ↪filename in os.listdir('submissions') if "submission" in filename]))
```

```
print("Last submission number:", last_submission_number)
print("Now creating submission number:", last_submission_number + 1)

# Create the new filename
new_filename = f'submission_{last_submission_number + 1}'

hello = os.environ.get('HELLO')
if hello is not None:
    new_filename += f'_{hello}'

print("New filename:", new_filename)
```

```
Last submission number: 77
Now creating submission number: 78
New filename: submission_78_jorge
```

[69]:
```
from autogluon.tabular import TabularDataset, TabularPredictor
train_data = TabularDataset('X_train_raw.csv')
train_data.drop(columns=['ds'], inplace=True)

label = 'y'
metric = 'mean_absolute_error'
time_limit = 60
presets = 'best_quality'
```

```
Loaded data from: X_train_raw.csv | Columns = 49 / 49 | Rows = 93024 -> 93024
```

[70]:
```
predictors = [None, None, None]
```

[71]:
```
loc = "A"
print(f"Training model for location {loc}...")
predictor = TabularPredictor(label=label, eval_metric=metric,␣
 ↪path=f"AutogluonModels/{new_filename}_{loc}").
 ↪fit(train_data[train_data["location"] == loc], time_limit=time_limit,␣
 ↪presets=presets)
predictors[0] = predictor
```

```
Warning: path already exists! This predictor may overwrite an existing
predictor! path="AutogluonModels/submission_78_jorge_A"
Presets specified: ['best_quality']
Stack configuration (auto_stack=True): num_stack_levels=1, num_bag_folds=8,
num_bag_sets=20
Beginning AutoGluon training … Time limit = 60s
AutoGluon will save models to "AutogluonModels/submission_78_jorge_A/"
AutoGluon Version:  0.8.1
Python Version:     3.10.12
Operating System:   Darwin
Platform Machine:   arm64
Platform Version:   Darwin Kernel Version 22.1.0: Sun Oct  9 20:15:09 PDT 2022;
```

```
root:xnu-8792.41.9~2/RELEASE_ARM64_T6000
Disk Space Avail:   34.50 GB / 494.38 GB (7.0%)
Train Data Rows:    34085
Train Data Columns: 47
Label Column: y
Preprocessing data …
AutoGluon infers your prediction problem is: 'regression' (because dtype of
label-column == float and many unique label-values observed).
        Label info (max, min, mean, stddev): (5733.42, 0.0, 630.59471,
1165.90242)
        If 'regression' is not the correct problem_type, please manually specify
the problem_type parameter during predictor init (You may specify problem_type
as one of: ['binary', 'multiclass', 'regression'])
Using Feature Generators to preprocess the data …
Fitting AutoMLPipelineFeatureGenerator…
        Available Memory:                    5172.74 MB
        Train Data (Original)  Memory Usage: 14.52 MB (0.3% of available memory)
        Inferring data type of each feature based on column values. Set
feature_metadata_in to manually specify special dtypes of the features.
        Stage 1 Generators:
                Fitting AsTypeFeatureGenerator…
                        Note: Converting 3 features to boolean dtype as they
only contain 2 unique values.
        Stage 2 Generators:
                Fitting FillNaFeatureGenerator…
        Stage 3 Generators:
                Fitting IdentityFeatureGenerator…
        Stage 4 Generators:
                Fitting DropUniqueFeatureGenerator…
        Stage 5 Generators:
                Fitting DropDuplicatesFeatureGenerator…
        Useless Original Features (Count: 1): ['location']
```

```
Training model for location A…
```

```
                These features carry no predictive signal and should be manually
investigated.
                This is typically a feature which has the same value for all
rows.
                These features do not need to be present at inference time.
        Unused Original Features (Count: 1): ['snow_drift:idx']
                These features were not used to generate any of the output
features. Add a feature generator compatible with these features to utilize
them.
                Features can also be unused if they carry very little
information, such as being categorical but having almost entirely unique values
or being duplicates of other features.
                These features do not need to be present at inference time.
                ('float', []) : 1 | ['snow_drift:idx']
```

```
        Types of features in original data (raw dtype, special dtypes):
                ('float', []) : 45 | ['absolute_humidity_2m:gm3',
'air_density_2m:kgm3', 'ceiling_height_agl:m', 'clear_sky_energy_1h:J',
'clear_sky_rad:W', …]
        Types of features in processed data (raw dtype, special dtypes):
                ('float', [])     : 43 | ['absolute_humidity_2m:gm3',
'air_density_2m:kgm3', 'ceiling_height_agl:m', 'clear_sky_energy_1h:J',
'clear_sky_rad:W', …]
                ('int', ['bool']) :  2 | ['elevation:m', 'snow_density:kgm3']
        0.2s = Fit runtime
        45 features in original data used to generate 45 features in processed
data.
        Train Data (Processed) Memory Usage: 11.79 MB (0.2% of available memory)
Data preprocessing and feature engineering runtime = 0.25s …
AutoGluon will gauge predictive performance using evaluation metric:
'mean_absolute_error'
        This metric's sign has been flipped to adhere to being higher_is_better.
The metric score can be multiplied by -1 to get the metric value.
        To change this, specify the eval_metric parameter of Predictor()
User-specified model hyperparameters to be fit:
{
        'NN_TORCH': {},
        'GBM': [{'extra_trees': True, 'ag_args': {'name_suffix': 'XT'}}, {},
'GBMLarge'],
        'CAT': {},
        'XGB': {},
        'FASTAI': {},
        'RF': [{'criterion': 'gini', 'ag_args': {'name_suffix': 'Gini',
'problem_types': ['binary', 'multiclass']}}, {'criterion': 'entropy', 'ag_args':
{'name_suffix': 'Entr', 'problem_types': ['binary', 'multiclass']}},
{'criterion': 'squared_error', 'ag_args': {'name_suffix': 'MSE',
'problem_types': ['regression', 'quantile']}}],
        'XT': [{'criterion': 'gini', 'ag_args': {'name_suffix': 'Gini',
'problem_types': ['binary', 'multiclass']}}, {'criterion': 'entropy', 'ag_args':
{'name_suffix': 'Entr', 'problem_types': ['binary', 'multiclass']}},
{'criterion': 'squared_error', 'ag_args': {'name_suffix': 'MSE',
'problem_types': ['regression', 'quantile']}}],
        'KNN': [{'weights': 'uniform', 'ag_args': {'name_suffix': 'Unif'}},
{'weights': 'distance', 'ag_args': {'name_suffix': 'Dist'}}],
}
AutoGluon will fit 2 stack levels (L1 to L2) …
Fitting 11 L1 models …
Fitting model: KNeighborsUnif_BAG_L1 … Training model for up to 39.82s of the
59.75s of remaining time.
        Not enough time to generate out-of-fold predictions for model. Estimated
time required was 467.0s compared to 51.74s of available time.
        Time limit exceeded… Skipping KNeighborsUnif_BAG_L1.
Fitting model: KNeighborsDist_BAG_L1 … Training model for up to 32.92s of the
```

```
52.85s of remaining time.
        Not enough time to generate out-of-fold predictions for model. Estimated
time required was 589.73s compared to 42.77s of available time.
        Time limit exceeded… Skipping KNeighborsDist_BAG_L1.
Fitting model: LightGBMXT_BAG_L1 … Training model for up to 24.22s of the
44.14s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -172.136         = Validation score   (-mean_absolute_error)
        17.93s   = Training   runtime
        42.81s   = Validation runtime
Completed 1/20 k-fold bagging repeats …
Fitting model: WeightedEnsemble_L2 … Training model for up to 59.75s of the
12.81s of remaining time.
        -172.136         = Validation score   (-mean_absolute_error)
        0.03s    = Training   runtime
        0.0s     = Validation runtime
Fitting 9 L2 models …
Fitting model: LightGBMXT_BAG_L2 … Training model for up to 12.77s of the
12.75s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -172.7621        = Validation score   (-mean_absolute_error)
        2.49s    = Training   runtime
        0.56s    = Validation runtime
Fitting model: LightGBM_BAG_L2 … Training model for up to 7.64s of the 7.63s
of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -171.8381        = Validation score   (-mean_absolute_error)
        2.02s    = Training   runtime
        0.19s    = Validation runtime
Fitting model: RandomForestMSE_BAG_L2 … Training model for up to 3.67s of the
3.67s of remaining time.
        -171.2924        = Validation score   (-mean_absolute_error)
        25.33s   = Training   runtime
        1.03s    = Validation runtime
Completed 1/20 k-fold bagging repeats …
Fitting model: WeightedEnsemble_L3 … Training model for up to 59.75s of the
-22.96s of remaining time.
        -169.7868        = Validation score   (-mean_absolute_error)
        0.15s    = Training   runtime
        0.0s     = Validation runtime
AutoGluon training complete, total runtime = 83.13s … Best model:
"WeightedEnsemble_L3"
TabularPredictor saved. To load, use: predictor =
TabularPredictor.load("AutogluonModels/submission_78_jorge_A/")
```

```
[72]: loc = "B"
print(f"Training model for location {loc}...")
predictor = TabularPredictor(label=label, eval_metric=metric,␣
 ↪path=f"AutogluonModels/{new_filename}_{loc}").
 ↪fit(train_data[train_data["location"] == loc], time_limit=time_limit,␣
 ↪presets=presets)
predictors[1] = predictor
```

```
Presets specified: ['best_quality']
Stack configuration (auto_stack=True): num_stack_levels=1, num_bag_folds=8,
num_bag_sets=20
Beginning AutoGluon training … Time limit = 60s
AutoGluon will save models to "AutogluonModels/submission_78_jorge_B/"
AutoGluon Version:  0.8.1
Python Version:     3.10.12
Operating System:   Darwin
Platform Machine:   arm64
Platform Version:   Darwin Kernel Version 22.1.0: Sun Oct  9 20:15:09 PDT 2022;
root:xnu-8792.41.9~2/RELEASE_ARM64_T6000
Disk Space Avail:   33.86 GB / 494.38 GB (6.8%)
Train Data Rows:    32844
Train Data Columns: 47
Label Column: y
Preprocessing data …
AutoGluon infers your prediction problem is: 'regression' (because dtype of
label-column == float and many unique label-values observed).
        Label info (max, min, mean, stddev): (1152.3, -0.0, 96.82478, 193.94649)
        If 'regression' is not the correct problem_type, please manually specify
the problem_type parameter during predictor init (You may specify problem_type
as one of: ['binary', 'multiclass', 'regression'])
Using Feature Generators to preprocess the data …
Fitting AutoMLPipelineFeatureGenerator…
        Available Memory:                    5099.57 MB
        Train Data (Original)  Memory Usage: 13.99 MB (0.3% of available memory)
        Inferring data type of each feature based on column values. Set
feature_metadata_in to manually specify special dtypes of the features.
        Stage 1 Generators:
                Fitting AsTypeFeatureGenerator…
                        Note: Converting 2 features to boolean dtype as they
only contain 2 unique values.
        Stage 2 Generators:
                Fitting FillNaFeatureGenerator…
        Stage 3 Generators:
                Fitting IdentityFeatureGenerator…
        Stage 4 Generators:
                Fitting DropUniqueFeatureGenerator…
        Stage 5 Generators:
                Fitting DropDuplicatesFeatureGenerator…
```

```
        Useless Original Features (Count: 1): ['location']
                These features carry no predictive signal and should be manually
investigated.
                This is typically a feature which has the same value for all
rows.
                These features do not need to be present at inference time.
        Types of features in original data (raw dtype, special dtypes):
                ('float', []) : 46 | ['absolute_humidity_2m:gm3',
'air_density_2m:kgm3', 'ceiling_height_agl:m', 'clear_sky_energy_1h:J',
'clear_sky_rad:W', …]
        Types of features in processed data (raw dtype, special dtypes):
                ('float', [])     : 44 | ['absolute_humidity_2m:gm3',
'air_density_2m:kgm3', 'ceiling_height_agl:m', 'clear_sky_energy_1h:J',
'clear_sky_rad:W', …]
                ('int', ['bool']) :  2 | ['elevation:m', 'snow_density:kgm3']
        0.1s = Fit runtime
        46 features in original data used to generate 46 features in processed
data.
        Train Data (Processed) Memory Usage: 11.63 MB (0.2% of available memory)
Data preprocessing and feature engineering runtime = 0.16s …
AutoGluon will gauge predictive performance using evaluation metric:
'mean_absolute_error'
        This metric's sign has been flipped to adhere to being higher_is_better.
The metric score can be multiplied by -1 to get the metric value.
        To change this, specify the eval_metric parameter of Predictor()
User-specified model hyperparameters to be fit:
{
        'NN_TORCH': {},
        'GBM': [{'extra_trees': True, 'ag_args': {'name_suffix': 'XT'}}, {},
'GBMLarge'],
        'CAT': {},
        'XGB': {},
        'FASTAI': {},
        'RF': [{'criterion': 'gini', 'ag_args': {'name_suffix': 'Gini',
'problem_types': ['binary', 'multiclass']}}, {'criterion': 'entropy', 'ag_args':
{'name_suffix': 'Entr', 'problem_types': ['binary', 'multiclass']}},
{'criterion': 'squared_error', 'ag_args': {'name_suffix': 'MSE',
'problem_types': ['regression', 'quantile']}}],
        'XT': [{'criterion': 'gini', 'ag_args': {'name_suffix': 'Gini',
'problem_types': ['binary', 'multiclass']}}, {'criterion': 'entropy', 'ag_args':
{'name_suffix': 'Entr', 'problem_types': ['binary', 'multiclass']}},
{'criterion': 'squared_error', 'ag_args': {'name_suffix': 'MSE',
'problem_types': ['regression', 'quantile']}}],
        'KNN': [{'weights': 'uniform', 'ag_args': {'name_suffix': 'Unif'}},
{'weights': 'distance', 'ag_args': {'name_suffix': 'Dist'}}],
}
AutoGluon will fit 2 stack levels (L1 to L2) …
Fitting 11 L1 models …
```

```
Fitting model: KNeighborsUnif_BAG_L1 … Training model for up to 39.88s of the
59.83s of remaining time.
```

```
Training model for location B…
```

```
        Not enough time to generate out-of-fold predictions for model. Estimated
time required was 715.24s compared to 51.82s of available time.
        Time limit exceeded… Skipping KNeighborsUnif_BAG_L1.
Fitting model: KNeighborsDist_BAG_L1 … Training model for up to 28.95s of the
48.9s of remaining time.
        Not enough time to generate out-of-fold predictions for model. Estimated
time required was 745.08s compared to 37.6s of available time.
        Time limit exceeded… Skipping KNeighborsDist_BAG_L1.
Fitting model: LightGBMXT_BAG_L1 … Training model for up to 17.56s of the
37.51s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -31.9466         = Validation score   (-mean_absolute_error)
        16.35s   = Training   runtime
        32.52s   = Validation runtime
Completed 1/20 k-fold bagging repeats …
Fitting model: WeightedEnsemble_L2 … Training model for up to 59.84s of the
14.94s of remaining time.
        -31.9466         = Validation score   (-mean_absolute_error)
        0.0s     = Training   runtime
        0.0s     = Validation runtime
Fitting 9 L2 models …
Fitting model: LightGBMXT_BAG_L2 … Training model for up to 14.93s of the
14.93s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -30.3331         = Validation score   (-mean_absolute_error)
        13.61s   = Training   runtime
        4.07s    = Validation runtime
Completed 1/20 k-fold bagging repeats …
Fitting model: WeightedEnsemble_L3 … Training model for up to 59.84s of the
-3.28s of remaining time.
        -30.3331         = Validation score   (-mean_absolute_error)
        0.0s     = Training   runtime
        0.0s     = Validation runtime
AutoGluon training complete, total runtime = 63.3s … Best model:
"WeightedEnsemble_L3"
TabularPredictor saved. To load, use: predictor =
TabularPredictor.load("AutogluonModels/submission_78_jorge_B/")
```

[73]:

```
loc = "C"
print(f"Training model for location {loc}...")
```

```python
predictor = TabularPredictor(label=label, eval_metric=metric,␣
↪path=f"AutogluonModels/{new_filename}_{loc}").
↪fit(train_data[train_data["location"] == loc], time_limit=time_limit,␣
↪presets=presets)
predictors[2] = predictor
```

```
Presets specified: ['best_quality']
Stack configuration (auto_stack=True): num_stack_levels=1, num_bag_folds=8,
num_bag_sets=20
Beginning AutoGluon training … Time limit = 60s
AutoGluon will save models to "AutogluonModels/submission_78_jorge_C/"
AutoGluon Version:  0.8.1
Python Version:     3.10.12
Operating System:   Darwin
Platform Machine:   arm64
Platform Version:   Darwin Kernel Version 22.1.0: Sun Oct  9 20:15:09 PDT 2022;
root:xnu-8792.41.9~2/RELEASE_ARM64_T6000
Disk Space Avail:   33.67 GB / 494.38 GB (6.8%)
Train Data Rows:    26095
Train Data Columns: 47
Label Column: y
Preprocessing data …
AutoGluon infers your prediction problem is: 'regression' (because dtype of
label-column == float and label-values can't be converted to int).
        Label info (max, min, mean, stddev): (999.6, -0.0, 77.63106, 165.81688)
        If 'regression' is not the correct problem_type, please manually specify
the problem_type parameter during predictor init (You may specify problem_type
as one of: ['binary', 'multiclass', 'regression'])
Using Feature Generators to preprocess the data …
Fitting AutoMLPipelineFeatureGenerator…
        Available Memory:                    5539.9 MB
        Train Data (Original)  Memory Usage: 11.12 MB (0.2% of available memory)
        Inferring data type of each feature based on column values. Set
feature_metadata_in to manually specify special dtypes of the features.
        Stage 1 Generators:
                Fitting AsTypeFeatureGenerator…
                        Note: Converting 3 features to boolean dtype as they
only contain 2 unique values.
        Stage 2 Generators:
                Fitting FillNaFeatureGenerator…
        Stage 3 Generators:
                Fitting IdentityFeatureGenerator…
        Stage 4 Generators:
                Fitting DropUniqueFeatureGenerator…
        Stage 5 Generators:
                Fitting DropDuplicatesFeatureGenerator…
        Useless Original Features (Count: 1): ['location']
```

```
Training model for location C…
                These features carry no predictive signal and should be manually
investigated.
                This is typically a feature which has the same value for all
rows.
                These features do not need to be present at inference time.
        Unused Original Features (Count: 1): ['snow_drift:idx']
                These features were not used to generate any of the output
features. Add a feature generator compatible with these features to utilize
them.
                Features can also be unused if they carry very little
information, such as being categorical but having almost entirely unique values
or being duplicates of other features.
                These features do not need to be present at inference time.
                ('float', []) : 1 | ['snow_drift:idx']
        Types of features in original data (raw dtype, special dtypes):
                ('float', []) : 45 | ['absolute_humidity_2m:gm3',
'air_density_2m:kgm3', 'ceiling_height_agl:m', 'clear_sky_energy_1h:J',
'clear_sky_rad:W', …]
        Types of features in processed data (raw dtype, special dtypes):
                ('float', [])     : 43 | ['absolute_humidity_2m:gm3',
'air_density_2m:kgm3', 'ceiling_height_agl:m', 'clear_sky_energy_1h:J',
'clear_sky_rad:W', …]
                ('int', ['bool']) :  2 | ['elevation:m', 'snow_density:kgm3']
        0.2s = Fit runtime
        45 features in original data used to generate 45 features in processed
data.
        Train Data (Processed) Memory Usage: 9.03 MB (0.2% of available memory)
Data preprocessing and feature engineering runtime = 0.22s …
AutoGluon will gauge predictive performance using evaluation metric:
'mean_absolute_error'
        This metric's sign has been flipped to adhere to being higher_is_better.
The metric score can be multiplied by -1 to get the metric value.
        To change this, specify the eval_metric parameter of Predictor()
User-specified model hyperparameters to be fit:
{
        'NN_TORCH': {},
        'GBM': [{'extra_trees': True, 'ag_args': {'name_suffix': 'XT'}}, {},
'GBMLarge'],
        'CAT': {},
        'XGB': {},
        'FASTAI': {},
        'RF': [{'criterion': 'gini', 'ag_args': {'name_suffix': 'Gini',
'problem_types': ['binary', 'multiclass']}}, {'criterion': 'entropy', 'ag_args':
{'name_suffix': 'Entr', 'problem_types': ['binary', 'multiclass']}},
{'criterion': 'squared_error', 'ag_args': {'name_suffix': 'MSE',
'problem_types': ['regression', 'quantile']}}],
```

```
        'XT': [{'criterion': 'gini', 'ag_args': {'name_suffix': 'Gini',
'problem_types': ['binary', 'multiclass']}}, {'criterion': 'entropy', 'ag_args':
{'name_suffix': 'Entr', 'problem_types': ['binary', 'multiclass']}},
{'criterion': 'squared_error', 'ag_args': {'name_suffix': 'MSE',
'problem_types': ['regression', 'quantile']}}],
        'KNN': [{'weights': 'uniform', 'ag_args': {'name_suffix': 'Unif'}},
{'weights': 'distance', 'ag_args': {'name_suffix': 'Dist'}}],
}
AutoGluon will fit 2 stack levels (L1 to L2) …
Fitting 11 L1 models …
Fitting model: KNeighborsUnif_BAG_L1 … Training model for up to 39.84s of the
59.78s of remaining time.
        Not enough time to generate out-of-fold predictions for model. Estimated
time required was 441.92s compared to 51.78s of available time.
        Time limit exceeded… Skipping KNeighborsUnif_BAG_L1.
Fitting model: KNeighborsDist_BAG_L1 … Training model for up to 31.34s of the
51.28s of remaining time.
        Not enough time to generate out-of-fold predictions for model. Estimated
time required was 425.96s compared to 40.72s of available time.
        Time limit exceeded… Skipping KNeighborsDist_BAG_L1.
Fitting model: LightGBMXT_BAG_L1 … Training model for up to 23.14s of the
43.08s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -18.3348         = Validation score   (-mean_absolute_error)
        20.95s   = Training   runtime
        37.42s   = Validation runtime
Completed 1/20 k-fold bagging repeats …
Fitting model: WeightedEnsemble_L2 … Training model for up to 59.78s of the
14.98s of remaining time.
        -18.3348         = Validation score   (-mean_absolute_error)
        0.0s     = Training   runtime
        0.0s     = Validation runtime
Fitting 9 L2 models …
Fitting model: LightGBMXT_BAG_L2 … Training model for up to 14.96s of the
14.95s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -18.9072         = Validation score   (-mean_absolute_error)
        3.2s     = Training   runtime
        0.62s    = Validation runtime
Fitting model: LightGBM_BAG_L2 … Training model for up to 9.19s of the 9.19s
of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -18.6441         = Validation score   (-mean_absolute_error)
        3.84s    = Training   runtime
        0.16s    = Validation runtime
```

```
Fitting model: RandomForestMSE_BAG_L2 … Training model for up to 3.41s of the
3.4s of remaining time.
```

2 Submit

```
[ ]: import pandas as pd
     import matplotlib.pyplot as plt

     train_data_with_dates = TabularDataset('X_train_raw.csv')
     train_data_with_dates["ds"] = pd.to_datetime(train_data_with_dates["ds"])

     test_data = TabularDataset('X_test_raw.csv')
     test_data["ds"] = pd.to_datetime(test_data["ds"])
     #test_data
```

```
Loaded data from: X_train_raw.csv | Columns = 49 / 49 | Rows = 93024 -> 93024
Loaded data from: X_test_raw.csv | Columns = 48 / 48 | Rows = 4608 -> 4608
```

```
[ ]: test_ids = TabularDataset('test.csv')
     test_ids["time"] = pd.to_datetime(test_ids["time"])
     # merge test_data with test_ids
     test_data_merged = pd.merge(test_data, test_ids, how="inner", right_on=["time",␣
      ↪"location"], left_on=["ds", "location"])

     #test_data_merged
```

```
Loaded data from: test.csv | Columns = 4 / 4 | Rows = 2160 -> 2160
```

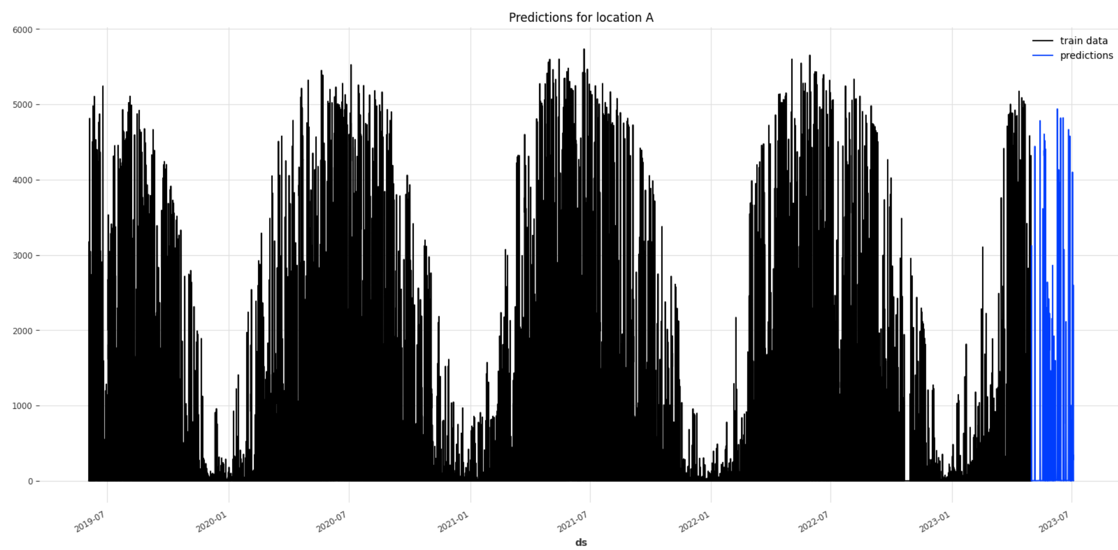
```
[ ]: # predict, grouped by location
     predictions = []
     location_map = {
         "A": 0,
         "B": 1,
         "C": 2
     }
     for loc, group in test_data.groupby('location'):
         i = location_map[loc]
         subset = test_data_merged[test_data_merged["location"] == loc].
      ↪reset_index(drop=True)
         #print(subset)
         pred = predictors[i].predict(subset)
         subset["prediction"] = pred
         predictions.append(subset)
```

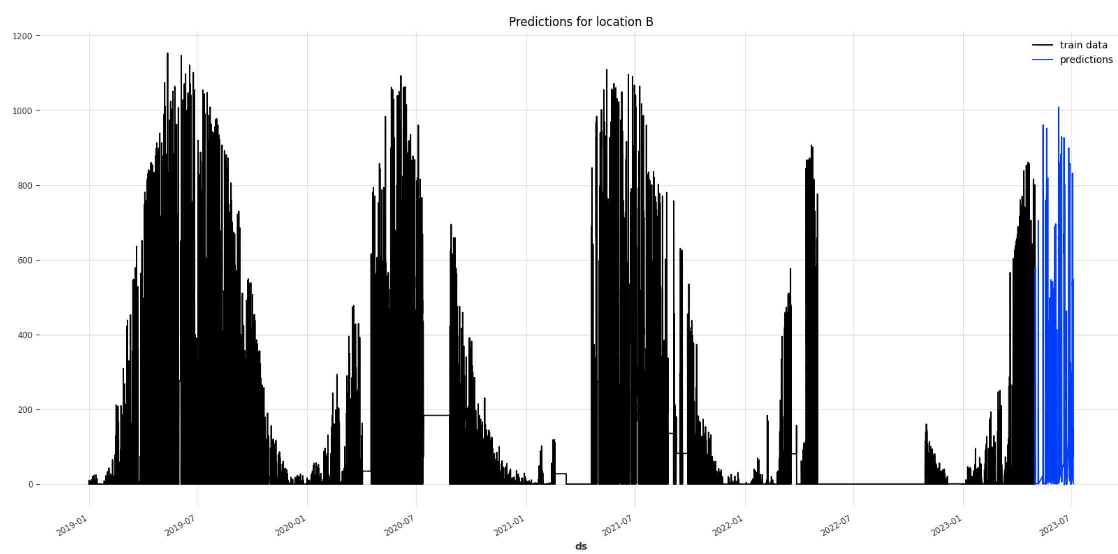
```
[ ]: # plot predictions for location A, in addition to train data for A
     for loc, idx in location_map.items():
         fig, ax = plt.subplots(figsize=(20, 10))
         # plot train data
```

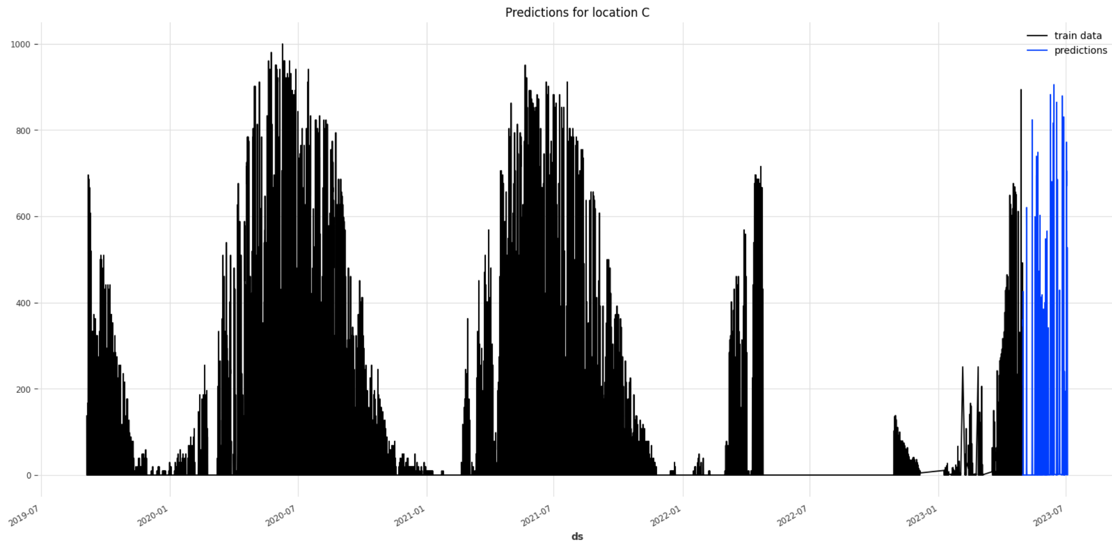
```
    train_data_with_dates[train_data_with_dates["location"]==loc].plot(x='ds', y='y', ax=ax, label="train data")

    # plot predictions
    predictions[idx].plot(x='ds', y='prediction', ax=ax, label="predictions")

    # title
    ax.set_title(f"Predictions for location {loc}")
```







[ ]:
```
# concatenate predictions
submissions_df = pd.concat(predictions)
submissions_df = submissions_df[["id", "prediction"]]
submissions_df
```

[ ]:
```
       id  prediction
0       0    2.846197
1       1    2.923443
2       2    3.005310
3       3   51.197372
4       4  266.546478
..    ...         ...
715  2155   93.187904
716  2156   85.414459
717  2157   31.074415
718  2158    4.270400
719  2159    1.171429

[2160 rows x 2 columns]
```

[ ]:
```
# Save the submission DataFrame to submissions folder, create new name based on
↪last submission, format is submission_<last_submission_number + 1>.csv

# Save the submission
print(f"Saving submission to submissions/{new_filename}.csv")
submissions_df.to_csv(os.path.join('submissions', f"{new_filename}.csv"),
↪index=False)
```

```
Saving submission to submissions/submission_71.csv
```

[ ]:
```
# feature importance
predictors[0].feature_importance(feature_stage="transformed")
```

[ ]:
```
# save this notebook to submissions folder
import subprocess
import os
subprocess.run(["jupyter", "nbconvert", "--to", "pdf", "--output", os.path.
↪join('notebook_pdfs', f"{new_filename}.pdf"), "autogluon_each_location.
↪ipynb"])
```

```
[NbConvertApp] Converting notebook autogluon_each_location.ipynb to pdf
[NbConvertApp] Support files will be in notebook_pdfs/submission_71_files/
[NbConvertApp] Making directory
./notebook_pdfs/submission_71_files/notebook_pdfs
[NbConvertApp] Writing 110734 bytes to notebook.tex
[NbConvertApp] Building PDF
[NbConvertApp] Running xelatex 3 times: ['xelatex', 'notebook.tex', '-quiet']
[NbConvertApp] Running bibtex 1 time: ['bibtex', 'notebook']
[NbConvertApp] WARNING | bibtex had problems, most likely because there were no
citations
[NbConvertApp] PDF successfully created
[NbConvertApp] Writing 272810 bytes to notebook_pdfs/submission_71.pdf
```

[ ]:
```
CompletedProcess(args=['jupyter', 'nbconvert', '--to', 'pdf', '--output',
'notebook_pdfs/submission_71.pdf', 'autogluon_each_location.ipynb'],
returncode=0)
```